REMARKS

Posture of the case

Claims 1-20 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Patent No. 5,797,013 ("Mahadevan") in view of Patent No. 5,862,384 ("Hirai") in view of Patent No. 6,748,589 ("Johnson").

Applicant submits amendments to claims 2 and 12, as set out herein above, to distinguish the present invention over the cited art and thereby overcome the rejection. Applicant herein cancels claims 1 and 11 and amends claims 7 and 17 to conform them to amended claims 2 and 12. Applicant herein amends claims 9 and 19 to correct a typographical error.

Brief discussion of certain features of the present invention

When a loop executes only a small number of times, such as, for example 0, 1 or 2 times, it may frequently occur that the code immediately before and after a loop body, i.e., the pre-loop code and post-loop code, will perform very little work. Present application, page 4, line 24 - page 5, line 1. According to the present invention, special cases of a loop executing a certain predetermined, small number of times are analyzed in compiling, and the compiler determines the effect of such cases on the pre-loop and post-loop code. Present application, page 8, lines 9 - 16. In addition to the features previously pointed out in the claims of the present application, the present invention also involves a recognition that this involves not only unrolling a loop body, which may lead to deleting code of the loop body or moving code from the loop body, but also actually performing certain computations during such compiling to determine values of one or more variables that govern termination of loop iterations, which that the present application refers to as "execution conditions." Present application, page 15, lines 22-25. Such an execution condition is obtained, as described in the present application, by the compiler considering the one or more variables that appear in a loop termination condition in the loop body, and computing a value or values for the one or more variables such that the value or values will terminate the loop in a predetermined number of iterations. Present application, page 10, lines 3-8. Then the compiler propagates such one or more values to the loop code body and pre-loop code and post-loop code, and detects resulting effects on all that code, which may include detecting redundancies or invariant expressions in the pre-loop and post-loop code so that the compiler may eliminate, or at least simplify, some of the pre-loop or post-loop code, as well as in the loop body. Present application, page 10, lines 3-8.

Amendments submitted herein

According to the above, amendments to claims 2 and 12 are submitted. Claim 2, for example now states that a predetermined number *n* is stored for a compiler and that in the non-optimized loop code segment the loop construct may be executed at run time at least a loop repetition number of times *n*, depending upon a value at run time of a variable in a loop termination condition of the loop code segment. Further, the amended claim states that a computation is performed by the compiler for determining *n+1* values for the variable in the loop termination condition, such that a first one of the values would, upon execution at run time, terminate the loop code segment in 0 iterations, a second one of the values would, upon execution at run time, terminate the loop code segment in 1 iteration, and an *(n+1)*th one of the values would, upon execution at run time, terminate the loop code segment in *n* iterations. The claim further states that the optimizing includes optimizing the non-optimized pre-loop, loop and post-loop code segments, which includes the compiler propagating the values to the loop code segment, the pre-loop code and the post-loop code. It also includes detecting effects on the loop code segment, the pre-loop code and the post-loop code of propagating the values, including detecting redundancies and invariant expressions in the loop code segment, the pre-loop code and the post-loop code resulting from the propagation of the values, so that the compiler may eliminate, or at least simplify, some of the loop code segment, pre-loop code and post-loop code to provide a consolidated code segment corresponding with conditions for execution of the loop said loop repetition number of times *n*. Thus, the claim states, the consolidated code includes certain code of the non-optimized loop code segment, the non-optimized pre-loop code and the non-optimized post-loop code and omits certain other code of the non-optimized loop code segment, the non-optimized pre-loop code and the non-optimized post-loop code. Claim 12 is amended in similar fashion, according to the form of the invention it claims.

No new matter is added in the amendments to claims, since the specification as originally submitted provides support, as described herein above.

Applicant contends that the above amendments patentably distinguish the invention. While Mahadevan discloses unrolling loops, neither Mahadevan, nor Mahadevan in combination with any of the cited art, teaches or suggests a compiler determining certain values for a variable in a loop termination condition, such that a first one of the values would, upon execution at run time, terminate the loop code segment in 0 iterations, a second one of the values would, upon execution at run time, terminate the loop code segment in 1 iteration, and so on, up to *n* iterations, and then propagating the computed values, as stated in the claims, so that the compiler may eliminate, or at least simplify, some of the loop code segment, pre-loop code and post-loop code to provide a consolidated code segment corresponding with conditions for execution of the loop said loop repetition number of times *n*, as now claimed.

Likewise, while Hirai discloses detecting and eliminating invariant expression, including in loops having function calls, it appears Hirai merely considers whether any arguments in an expression correspond to any arguments of function calls in a loop. Hirai does not teach or suggest, neither alone nor in combination with any of the cited art, a compiler determining certain values for a variable in a loop termination condition, such that a first one of the values would, upon execution at run time, terminate the loop code segment in 0 iterations, a second one of the values would, upon execution at run time, terminate the loop code segment in 1 iteration, and so on, up to *n* iterations, and then propagating the computed values, as stated in the claims, so that the compiler may eliminate, or at least simplify, some of the loop code segment, pre-loop code and post-loop code to provide a consolidated code segment corresponding with conditions for execution of the loop said loop repetition number of times *n*, as now claimed.

Johnson discloses state-saving hardware, including a buffer, in a microprocessor for rolling back or committing execution to permit speculative execution, and discloses using a software scheduler to reorder translated instructions before executing them in sequences, and taking advantage of the state-saving hardware to implement more aggressive reordering, Johnson also discloses creating a duplicate set of translated instructions to use for rolling back, should that become necessary. However, Johnson, does not teach or suggest, neither alone nor in combination with any of the cited art, a compiler determining certain values for a variable in a loop termination condition, such that a first one of the values would, upon execution at run time, terminate the loop code segment in 0 iterations, a second one of the values would, upon execution at run time, terminate the loop code segment in 1 iteration, and so on, up to *n* iterations, and then propagating the computed values, as stated in the claims, so that the compiler may eliminate, or at least simplify, some of the loop code segment, pre-loop code and post-loop code to provide a consolidated code segment corresponding with conditions for execution of the loop said loop repetition number of times *n*, as now claimed.

For the above reasons, Applicant contends independent claims 2 and 12 of the present application, as amended, are allowable. Applicant also contends that dependent claims 7-9, and 17-19 are patentably distinct at least because they each depend on respectively allowable independent claims. MPEP 2143.03.

PRIOR ART OF RECORD

Applicant has reviewed the prior art of record cited by but not relied upon by Examiner, and asserts that the invention is patentably distinct.

REQUESTED ACTION

Applicant contends that the invention as claimed in accordance with amendments submitted herein is patentably distinct, and hereby requests that Examiner grant allowance and prompt passage of the application to issuance.

Respectfully submitted,

Anthony VS England

Anthony V. S. England
Attorney for Applicants
Registration No. 35,129
512-477-7165
a@aengland.com